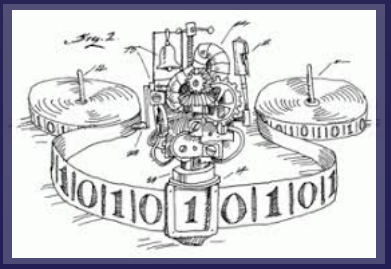
Terminale spécialité Numérique et Sciences Informatiques
Leçons

Laurent Brisepierre

27 août 2024



Sommaire

Chapitre 1

Reprise en main de Python

I Des rappels théoriques

1 Affecter une variable

Interprétation	Pseudo-code	Python
x prend la valeur 3	$x \leftarrow 3$	$x = 3$

Les types usuels des valeurs manipulées en Python sont :

- Les entiers (type int)
- Les flottants (type float, assimilables aux décimaux)
- Les booléens (type bool) de valeurs True ou False, résultats d'un test
- Les chaînes de caractères (type str) qui représentent les textes
- Les listes (type list) qui sont des séquences d'éléments

2 Opérations sur les nombres

Addition	$a+b$
Soustraction	$a-b$
Multiplication	$a*b$
Puissance correspond à l'opération a^b	$a**b$

Quotient de la division euclidienne partie entière du quotient de a par b.	$a//b$
Division valeur approchée de a divisé par b.	a/b
Reste de la division euclidienne modulo	$a\%b$

Ces opérations sont valables pour les int et les float. Les priorités opératoires restent valables en Python, les parenthèses « (» et «) » modifient ces priorités.

3 Définir une fonction

Interprétation	Pseudo-code	Python

f(x) = 2 x + 3	Définir f (x)
	  Renvoyer 2 x + 3
		def f(x):
		  return 2*x+3

Attention, les deux points et l'indentation sont indispensables...

4 Effectuer un test

Interprétation	Pseudo-code	Python
x est-il égal à y ?	$x = y$	$x == y$
x est-il distinct de y ?	$x \neq y$	$x! = y$
x est-il strictement supérieur à y ?	$x > y$	$x > y$
x est-il supérieur ou égal à y ?	$x \geqslant y$	$x >= y$
x est-il strictement inférieur à y ?	$x < y$	$x < y$
x est-il inférieur ou égal à y ?	$x \leqslant y$	$x <= y$
x est-il dans y ?	$x \in y$	x in y
x est-il hors de y ?	$x \notin y$	x not in y

5 Instructions conditionnelle

Remarque	Pseudo-code	Python

La partie « sinon » est optionnelle	Si A > 0 Alors
	  A → -A
	Sinon
	  A → f(A)
	Fin Si
		if A > 0:
		  A = -A
		else:
		  A = f(A)

À nouveau, les deux points et l'indentation sont indispensables. S'il y a plusieurs instructions imbriquées, on peut utiliser elif, contraction de « else if ».

6 Boucle itérateur

Pseudo-code	Python	Entiers parcourus

Répéter 20 fois
  A ← A+2
Fin répétition
	for i in range(20):
	  A = A+2
		De 0 à 19 …

Pour i allant de 1 à N :
  A ← A+i
Fin Pour
	for i in range(1, N+1):
	  A = A + i
		De 1 à N …

`range(d, f, p)` génère les entiers de d (inclus) à f (exclus), avec un pas de p. Par défaut, d = 0 et p = 1 donc écrire `range(f)` signifie `range(0, f, 1)`.

7 Boucle conditionnelle

Remarque	Pseudo-code	Python

Lorsqu'on passe sur machine, attention aux « boucles infinies »...	Tant que n > 1
	  n ← reste de n/q
	Fin Tant Que
		while n > 1:
		  n = n%q

II Des pratiques à connaître

1 Parcours de listes et de chaînes de caractères

En Python, il y a deux méthodes pour parcourir les listes (tableaux) et les textes (chaines de caractères) : **parcours valeur par valeur** ou **parcours par indice des valeurs.**

Lors d'un **parcours par valeur**, on n'a pas accès à l'indice (la position de la valeur) mais uniquement à la valeur.
Pour choisir entre les deux, il faut donc se poser la question suivante :
Est-ce que j'ai besoin de connaître l'indice de la valeur que je regarde ?
Si la réponse est oui, il faut faire un parcours par indice.
Si la réponse est non, vous pouvez utiliser un parcours par valeur (mais un parcours par indice fonctionnerait aussi).

- Parcours valeur par valeur :
  Exemple d'un code Python :

```
def parcours_par_valeur(donnees):
    for val in donnees:
        print(val)
```

On obtient :

```
>>> parcours_par_valeur([3, -2, 4])
3
```

```
-2
4
>>> parcours_par_valeur("AVION")
A
V
I
O
N
```

- Parcours par indice :
  Exemple d'un code Python :

```
def parcours_par_indice(donnees):
    for i in range(len(donnees)):
        print(i, donnees[i])  # i représente la position d'une valeur et donnees[i] la valeur correspon
```

On obtient :

```
>>> parcours_par_indice([3, -2, 4])
0 3
1 -2
2 4
>>> parcours_par_indice("AVION")
0 A
1 V
2 I
3 O
4 N
```

2 Les dictionnaires

a Rappels

Un **dictionnaire** est une structure Python qui permet de stocker des données par paires **clé : valeur**.

Il s'écrit avec des accolades :
D = clé1 : valeur1 , clé2 : valeur 2, . . .

Il n'y a pas d'ordre dans un dictionnaire.
On accède à une valeur d'un dictionnaire à l'aide de sa clé correspondante.

La syntaxe **len(mon_dico)** renvoie le nombre de **couple** (cle, valeur) du dictionnaire.

Le parcours avec la boucle **for element in mon_dico** permet de parcourir les clés de **mon_dico**

La syntaxe **mon_dico[cle]** permet d'accéder à la valeur de la clé correspondante.

b Un exemple

On considère 4 dictionnaires qui contiennent chacun les caractéristiques de 4 Pokémons différents :

```
pokemon1 = {'Nom' : 'Carapuce', 'HP' : 44, 'Attaque' : 48, 'Defense' : 65, 'Vitesse' : 43, 'Type' : ['Eau']
pokemon2 = {'Nom' : 'Evoli', 'HP' : 55, 'Attaque' : 55, 'Defense' : 50, 'Vitesse' : 55, 'Type' : ['Normal']
pokemon3 = {'Nom' : 'Salameche', 'HP' : 39, 'Attaque' : 52, 'Defense' : 43, 'Vitesse' : 65, 'Type' : ['Feu'
pokemon4 = {'Nom' : 'Reptencil', 'HP' : 58, 'Attaque' : 64, 'Defense' : 58, 'Vitesse' : 80, 'Type' : ['Eau'
```

Ces dictionnaires sont regroupés dans une liste :

```
liste_pokemon = [pokemon1, pokemon2, pokemon3, pokemon4]
```

On souhaite écrire un **script** (fonction Python) **pokemons_rapides()** qui renverra une liste contenant les noms des Pokémons dont la vitesse est supérieure à 60.
On doit obtenir l'affichage : **['Salameche', 'Reptencil']** ou **['Reptencil', 'Salameche']**.

Le principe de l'algorithme est le suivant :

- On commence par parcourir la liste **liste_pokemon** pour étudier les Pokémons les uns après les autres.
- Les informations de chaque Pokémons sont contenues dans un dictionnaire. Il faut chercher la clé **'Vitesse'** de ce dictionnaire pour étudier sa valeur.
- Si cette vitesse est supérieur à 60, on doit stocker le nom de ce Pokémon dans la liste finale qui sera renvoyé par la fonction à la fin.
  Il faut donc initialiser cette liste, qui n'existait pas au début de l'algorithme.
  Le nom du Pokémon sera accessible grâce à la clé **"Nom"**.

En Python :

```
pokemon1 = {'Nom' : 'Carapuce', 'HP' : 44, 'Attaque' : 48, 'Defense' : 65, 'Vitesse' : 43, 'Type' : ['Eau']
pokemon2 = {'Nom' : 'Evoli', 'HP' : 55, 'Attaque' : 55, 'Defense' : 50, 'Vitesse' : 55, 'Type' : ['Normal']
pokemon3 = {'Nom' : 'Salameche', 'HP' : 39, 'Attaque' : 52, 'Defense' : 43, 'Vitesse' : 65, 'Type' : ['Feu'
pokemon4 = {'Nom' : 'Reptencil', 'HP' : 58, 'Attaque' : 64, 'Defense' : 58, 'Vitesse' : 80, 'Type' : ['Eau'

liste_pokemon = [pokemon1, pokemon2, pokemon3, pokemon4]

def pokemons_rapides() :

    L = []

    for pokemon in liste_pokemon:
        if pokemon['Vitesse'] > 60 :
            nom_pokemon = pokemon['Nom']
            L.append(nom_pokemon)

    return(L)
```

Puis dans la console, on obtient :

```
>>> pokemons_rapides()
['Salameche', 'Reptencil']
```